Asynchronous and Maximally Parallel Deterministic Controlled Non-Cooperative P Systems Characterize $NFIN \cup coNFIN$

Artiom Alhazov[1,2] and Rudolf Freund[3]

[1] Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Viale Sarca 336, 20126 Milano, Italy
artiom.alhazov@unimib.it
[2] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Academiei 5, Chişinău MD-2028 Moldova
artiom@math.md
[3] Faculty of Informatics, Vienna University of Technology
Favoritenstr. 9, 1040 Vienna, Austria
E-mail: rudi@emcc.at

Summary. Membrane systems (with symbol objects) are distributed controlled multiset processing systems. Non-cooperative P systems with either promoters or inhibitors (of weight not restricted to one) are known to be computationally complete. In this paper we show that the power of the deterministic subclass of such systems is computationally complete in the sequential mode, but only subregular in the asynchronous mode and in the maximally parallel mode.

1 Introduction

The most famous membrane computing model where determinism is a criterion of universality versus decidability is the model of catalytic P systems, see [2] and [4].

It is also known that non-cooperative rewriting P systems with either promoters or inhibitors are computationally complete, [1]. Moreover, the proof satisfies some additional properties:

- Either promoters of weight 2 or inhibitors of weight 2 are enough.
- The system is non-deterministic, but it restores the previous configuration if the guess is wrong, which leads to correct simulations with probability 1.

The purpose of this paper is to formally prove that computational completeness cannot be achieved by deterministic systems when working in the asynchronous or in the maximally parallel mode.

2 Definitions

An *alphabet* is a finite non-empty set V of abstract *symbols*. The free monoid generated by V under the operation of concatenation is denoted by V^*; the *empty string* is denoted by λ, and $V^* \setminus \{\lambda\}$ is denoted by V^+. The set of non-negative integers is denoted by $\mathbb{N}$; a set S of non-negative integers is called *co-finite* if $\mathbb{N} \setminus S$ is finite. The family of all finite (co-finite) sets of non-negative integers is denoted by $NFIN$ ($coNFIN$, respectively). The family of all recursively enumerable sets of non-negative integers is denoted by NRE. In the following, we will use $\subseteq$ both for the subset as well as the submultiset relation.

Since flattening the membrane structure of a membrane system preserves both determinism and the model, in the following we restrict ourselves to consider membrane systems as one-region multiset rewriting systems.

A *(one-region) membrane system (P system)* is a tuple

$$\Pi = (O, \Sigma, w, R'),$$

where O is a finite alphabet, $\Sigma \subseteq O$ is the input sub-alphabet, $w \in O^*$ is a string representing the initial multiset, and R' is a set of rules of the form $r : u \to v$, $u \in O^+$, $v \in O^*$.

A configuration of the system Π is represented by a multiset of objects from O contained in the region, the set of all configurations over O is denoted by $\mathbb{C}(O)$. A rule $r : u \to v$ is applicable if the current configuration contains the multiset specified by u. Furthermore, applicability may be controlled by *context conditions*, specified by pairs of sets of multisets.

Definition 1. *A* rule with context conditions $(r, (P_1, Q_1), \cdots, (P_m, Q_m))$ *is applicable to a configuration C if r is applicable, and there exists some $j \in \{1, \cdots, m\}$ for which*

- *there exists some $p \in P_j$ such that $p \subseteq C$ and*
- *$q \not\subseteq C$ for all $q \in Q_j$.*

In words, context conditions are satisfied if there exists a pair of sets of multisets (called *promoter set* and *inhibitor set*, respectively), such that at least one multiset in the promoter set is a submultiset of the current configuration, and no multiset in the inhibitor set is a submultiset of the current configuration.

Definition 2. *A* P system with context conditions and priorities on the rules *is a construct*

$$\Pi = (O, \Sigma, w, R', R, >)$$

where (O, Σ, w, R') is a (one-region) P system as defined above, R is a set of rules with context conditions and $>$ is a priority relation on the rules in R; if rule r' has priority over rule r, denoted by $r' > r$, then r cannot be applied if r' is applicable.

Throughout the paper, we will use the word *control* to mean that at least one of these features is allowed (context conditions or promoters or inhibitors only and eventually priorities).

In the *sequential mode* (*sequ*), a computation step consists in the non-deterministic application of one applicable rule r, replacing its left-hand side ($lhs\,(r)$) with its right-hand side ($rhs\,(r)$). In the *maximally parallel mode* (*maxpar*), multiple applicable rules may be chosen non-deterministically to be applied in parallel to the underlying configuration to disjoint submultisets, possibly leaving some objects idle, under the condition that no further rule is applicable to them. In the *asynchronous mode* (*asyn*), any positive number of applicable rules may be chosen non-deterministically to be applied in parallel to the underlying configuration to disjoint submultisets. The computation step between two configurations C and C' is denoted by $C \Rightarrow C'$, thus yielding the binary relation $\Rightarrow: \mathbb{C}\,(O) \times \mathbb{C}\,(O)$. A computation halts when there are no rules applicable to the current configuration (*halting configuration*) in the corresponding mode.

The computation of a *generating* P system starts with w, and its result is $|x|$ if it halts, an *accepting* system starts with wx, $x \in \Sigma^*$, and we say that $|x|$ is its results – is accepted – if it halts. The set of numbers generated/accepted by a P system working in the mode α is the set of results of its computations for all $x \in \Sigma^*$ and denoted by $N_g^{\alpha}(\Pi)$ and $N_a^{\alpha}(\Pi)$, respectively. The family of sets of numbers generated/accepted by a family of (one-region) P systems with context conditions and priorities on the rules with rules of type β working in the mode α is denoted by $N_{\delta}OP_1^{\alpha}\left(\beta, \left(pro_{k,l}, inh_{k',l'}\right)_d, pri\right)$ with $\delta = g$ for the generating and $\delta = a$ for the accepting case; d denotes the maximal number m in the rules with context conditions $(r, (P_1, Q_1), \cdots, (P_m, Q_m))$; k and k' denote the maximum number of promoters/inhibitors in the P_i and Q_i, respectively; l and l' indicate the maximum of weights of promoters and inhibitors, respectively. If any of these numbers k, k', l, l' is not bounded, we replace it by $*$. As types of rules we are going to distinguish between cooperative ($\beta = coo$) and non-cooperative (i.e., the left-hand side of each rule is a single object; $\beta = ncoo$) ones.

In the case of accepting systems, we also consider the idea of determinism, which means that in each step of any computation at most one (multiset of) rule(s) is applicable; in this case, we write *deta* for δ.

In the literature, we find a lot of restricted variants of P systems with context conditions and priorities on the rules, e.g., we may omit the priorities or the context conditions completely. If in a rule $(r, (P_1, Q_1), \cdots, (P_m, Q_m))$ we have $m = 1$, we say that $(r, (P_1, Q_1))$ is a rule with a *simple context condition*, and we omit the inner parentheses in the notation. Moreover, context conditions only using promoters are denoted by $r|_{p_1,\cdots,p_n}$, meaning $(r, \{p_1, \cdots, p_n\}, \emptyset)$, or, equivalently, $(r, (p_1, \emptyset), \cdots, (p_n, \emptyset))$; context conditions only using inhibitors are denoted by $r|_{\neg q_1,\cdots,\neg q_n}$, meaning $(r, \lambda, \{q_1, \cdots, q_n\})$, or $r|_{\neg\{q_1,\cdots,q_n\}}$. Likewise, a rule with both promoters and inhibitors can be specified as a rule with a simple context condition, i.e., $r|_{p_1,\cdots,p_n,\neg q_1,\cdots,\neg q_n}$ stands for $(r, \{p_1, \cdots, p_n\}, \{q_1, \cdots, q_n\})$. Finally, promoters and inhibitors of weight one are called *atomic*.

Remark 1. If we do not consider determinism, then (the effect of) the rule $(r, (P_1, Q_1), \cdots, (P_m, Q_m))$ is equivalent to (the effect of) the collection of rules $\{(r, P_j, Q_j) \mid 1 \leq j \leq m\}$, no matter in which mode the P system is working (obviously, the priority relation has to be adapted accordingly, too).

Remark 2. Let $(r, \{p_1, \cdots, p_n\}, Q)$ be a rule with a simple context condition; then we claim that (the effect of) this rule is equivalent to (the effect of) the collection of rules

$$\{(r, \{p_j\}, Q \cup \{p_k \mid 1 \leq k < j\}) \mid 1 \leq j \leq m\}$$

even in the the case of a deterministic P system: If the first promoter is chosen to make the rule r applicable, we do not care about the other promoters; if the second promoter is chosen to make the rule r applicable, we do not allow p_1 to appear in the configuration, but do not care about the other promoters p_3 to p_m; in general, when promoter p_j is chosen to make the rule r applicable, we do not allow p_1 to p_{j-1} to appear in the configuration, but do not care about the other promoters p_{j+1} to p_m; finally, we have the rule $\{(r, \{p_m\}, Q \cup \{p_k \mid 1 \leq k < m\})\}$. If adding $\{p_k \mid 1 \leq k < j\}$ to Q has the effect of prohibiting the promotor p_j from enabling the rule r to be applied, this makes no harm as in this case one of the promoters p_k, $1 \leq k < j$, must have the possibility for enabling r to be applied. By construction, the domains of the new context conditions now are disjoint, so this transformation does not create (new) non-determinism. In a similar way, this transformation may be performed on context conditions which are not simple. Therefore, without restricting generality, the set of promoters may be assumed to be a singleton. In this case, we may omit the braces of the multiset notation for the promoter multiset and write (r, p, Q).

Example 1. Consider an arbitrary finite set H of numbers. Choose $K = \max(H) + 1$; then we construct the following deterministic accepting P system with promoters and inhibitors:

$$\begin{aligned}
\Pi &= (O, \{a\}, s_0 f_0 \cdots f_K, R', R), \\
O &= \{a\} \cup \{s_i, f_i \mid 0 \leq i \leq K\}, \\
R' &= \{s_i \to s_{i+1} \mid 0 \leq i \leq K-1\} \cup \{f_i \to f_i \mid 0 \leq i \leq K\}, \\
R &= \{s_i \to s_{i+1}|_{a^{i+1}}, \mid 0 \leq i \leq K-1\} \\
&\quad \cup \left\{f_i \to f_i|_{s_i, \neg a^{i+1}}, \mid 0 \leq i < K,\ i \notin H\right\} \cup \{f_K \to f_K|_{s_K}\}.
\end{aligned}$$

The system step by step, by the application of the rule $s_i \to s_{i+1}|_{a^{i+1}}$, $0 \leq i < K$, checks if (at least) $i+1$ copies of the symbol a are present. If the computation stops after i steps, i.e., if the input has consisted of exactly i copies of a, then this input is accepted if and only if $i \in H$, as exactly in this case the system does not start an infinite loop with using $f_i \to f_i|_{s_i, \neg a^{i+1}}$. If the input has contained more than $\max(H)$ copies of a, then the system arrives in the state s_K and will loop forever with $f_K \to f_K|_{s_K}$. Therefore, exactly H is accepted. To accept the complement of H instead, we simply change $i \notin H$ to $i \in H$ and as well omit the rule $f_K \to f_K|_{s_K}$. It is easy to see that for the maximally parallel mode, we can

replace each rule $f_i \to f_i|_{s_i,\neg a^{i+1}}$ by the corresponding rule $f_i \to f_i|_{s_i}$; in this case, this rule may be applied with still some a being present while the system passes through the state s_i, but it will not get into an infinite loop in that case.

In sum, we have shown that

$$N_{deta}OP_1^{asyn}\left(ncoo,(pro_{1,*},inh_{1,*})_1\right) \supseteq FIN \cup coNFIN$$

and

$$N_{deta}OP_1^{maxpar}\left(ncoo,pro_{1,*}\right) \supseteq FIN \cup coNFIN.$$

Example 2. For P systems working in the maximally parallel way we can even construct a system with inhibitors only:

$$\begin{aligned}
\Pi &= (O,\{a\},ts_K,R),\\
O &= \{a,t\} \cup \{s_i \mid 0 \le i \le K\},\\
R' &= \{s_i \to ts_{i-1}, s_i \to s_i \mid 1 \le i \le K\} \cup \{t \to \lambda, s_0 \to s_0\},\\
R &= \{s_i \to ts_{i-1}|_{\neg a^i} \mid 1 \le i \le K\}\\
&\quad \cup \{t \to \lambda\} \cup \{s_i \to s_i|_{\neg t} \mid 0 \le i \le K,\ i \notin H\}.
\end{aligned}$$

This construction does not carry over to the case of the asynchronous mode, as the rule $t \to \lambda$ is applied in parallel to the rules $s_i \to ts_{i-1}|_{\neg a^i}$ until the input a^i is reached. In this case, the system canot change the state s_i anymore, and then it starts to loop if and only if $i \notin H$. To accept the complement of H instead, change $i \in H$ to $i \notin H$, i.e., in sum, we have proved that

$$N_{deta}OP_1^{maxpar}\left(ncoo,inh_{1,*}\right) \supseteq FIN \cup coNFIN.$$

As we shall show later, all the inclusions stated in Example 1 and Example 2 are equalities.

Remark 3. As in a P system $(O,\Sigma,w,R',R,>)$ the set of rules R' can easily be deduced from the set of rules with context conditions R, we omit R' in the description of the P system. Moreover, for systems having only rules with a simple context condition, we omit d in the description of the families of sets of numbers and simply write

$$N_\delta OP_1^\alpha\left(\beta,pro_{k,l},inh_{k',l'},pri\right).$$

Moreover, each control mechanism not used can be omitted, e.g., if no priorities and only promoters are used, we only write $N_\delta OP_1^\alpha\left(\beta,pro_{k,l}\right)$.

2.1 Register machines

In what follows we will need to simulate register machines; here we briefly recall their definition and some of their computational properties. A *register machine* is a tuple $M = (m,B,l_0,l_h,P)$, where m is the number of registers, P is the set of instructions bijectively labeled by elements of B, $l_0 \in B$ is the initial label, and $l_h \in B$ is the final label. The instructions of M can be of the following forms:

- $l_1 : (ADD(j), l_2, l_3)$, with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq m$.
  Increase the value of register j by one, and non-deterministically jump to instruction l_2 or l_3. This instruction is usually called *increment*.
- $l_1 : (SUB(j), l_2, l_3)$, with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq m$.
  If the value of register j is zero then jump to instruction l_3, otherwise decrease the value of register j by one and jump to instruction l_2. The two cases of this instruction are usually called *zero-test* and *decrement*, respectively.
- $l_h : HALT$. Stop the execution of the register machine.

A register machine is *deterministic* if $l_2 = l_3$ in all its ADD instructions. A *configuration* of a register machine is described by the contents of each register and by the value of the program counter, which indicates the next instruction to be executed. Computations start by executing the first instruction of P (labeled with l_0), and terminate with reaching a $HALT$-instruction.

Register machines provide a simple universal computational model [5]. We here consider register machines used as *accepting* or as *generating* devices. In accepting register machines, a vector of non-negative integers is accepted if and only if the register machine halts having it as input. Usually, without loss of generality, we may assume that the instruction $l_h : HALT$ always appears exactly once in P, with label l_h. In the generative case, we start with empty registers and take the results of all possible halting computations.

3 Results

In this section we mainly investigate deterministic accepting P systems with context conditions and priorities on the rules (*deterministic P systems* for short) using only non-cooperative rules and working in the sequential, the asynchronous, and the maximally parallel mode.

Remark 4. We first notice that maximal parallelism in systems with non-cooperative rules means the total parallelism for all symbols to which at least one rule is applicable, and determinism guarantees that "at least one" is "exactly one" for all reachable configurations and objects. Determinism in the sequential mode requires that at most one symbol has an associated applicable rule for all reachable configurations. Surprisingly enough, in the case of the asynchronous mode we face an even worse situation than in the case of maximal parallelism – if more than one copy of a specific symbol is present in the configuration, then no rule can be applicable to such a symbol in order not to violate the condition of determinism.

We now define the *bounding* operation over multisets, with a parameter $k \in \mathbb{N}$ as follows:

$$\text{for } u \in O^*,\ b_k(u) = v \text{ with } |v|_a = \min(|u|_a, k) \text{ for all } a \in O.$$

The mapping b_k "crops" the multisets by removing copies of every object a present in more than k copies until exactly k remain. For two multisets u, u', $b_k(u) = b_k(u')$ if for every $a \in O$, either $|u|_a = |u'|_a < k$, or $|u|_a \geq k$ and $|u'|_a \geq k$. Mapping b_k induces an equivalence relation, mapping O^* into $(k+1)^{|O|}$ equivalence classes. Each equivalence class corresponds to specifying, for each $a \in O^*$, whether no copy, one copy, or ... $k-1$ copies, or "k copies or more" are present. We denote the range of b_k by $\{0, \cdots, k\}^O$.

Lemma 1. *Context conditions are equivalent to predicates defined on boundings.*

Proof. We start by representing context conditions by predicates on boundings. Consider a rule with a simple context condition (r, p, Q), and let the current configuration be C. Then, it suffices to take $k \geq \max(|p|, \max\{|q| \mid q \in Q\})$, and let $C' = b_k(C)$. The applicability condition for (r, p, Q) may be expressed as $p \subseteq C' \wedge \left(\bigwedge_{q \in Q} q \not\subseteq C'\right)$. Indeed, $x \subseteq C \longleftrightarrow x \subseteq C'$ for every multiset x with $|x| \leq k$, because for every $a \in O$, $|x|_a \leq |C|_a \longleftrightarrow |x|_a \leq \min(|C|_a, k)$ holds if $|x|_a \leq k$. Finally, we notice that context conditions which are not simple can be represented by a disjunction of the corresponding predicates.

Conversely, we show that any predicate $E \subseteq \{0, \cdots, k\}^O$ for the bounding mapping b_k for rule r can be represented by some context conditions. For each multiset $c \in E$, we construct a simple context condition to the effect of "contains c, but, for each a contained in c for less than k times, not more than $|c|_a$ symbols a":

$$\left\{\left(r, c, \left\{a^{|c|_a+1} \mid |c|_a < k\right\}\right) \mid c \in E\right\}.$$

Joining multiple simple context conditions over the same rule into one rule with context conditions concludes the proof. □

The following theorem is valid even when the rules are not restricted to non-cooperative ones, and when determinism is not required, in either derivation mode (also see [3]).

Theorem 1. *Priorities are subsumed by conditional contexts.*

Proof. A rule is prohibited from being applicable due to a priority relation if and only if at least one of the rules with higher priority might be applied. Let r be a rule of a P system $(O, \Sigma, w, R', R, >)$, and let $r_1 > r, \cdots, r_n > r$. Hence, the rule r is not blocked by the rules $r_1, \cdots, r_n$ if and only if the left-hand sides of the rules $r_1, \cdots, r_n, lhs(r_1), \cdots, lhs(r_n)$ are not present in the current configuration or the context conditions given in these rules are not fulfilled. According to Lemma 1, these context conditions can be formulated as predicates on the bounding b_k where k is the maximum of weights of all left-hand sides, promoters, and inhibitors in the rules with higher priority $r_1, \cdots, r_n$. Together with the context conditions from r itself, we finally get context conditions for a new rule r' simulating r, but also incorporating the conditions of the priority relation. Performing this transformation for all rules r concludes the proof. □

Remark 5. From [3] we already know that in the case of rules without context conditions, the context conditions in the new rules are only sets of atomic inhibitors, which also follows from the construction given above. A careful investigation of the construction given in the proof of Theorem 1 reveals the fact that the maximal weights for the promoters and inhibitors to be used in the new system are bounded by the number k in the bounding b_k.

3.1 Sequential Systems

Although throughout the rest of the paper we are not dealing with sequential systems anymore, the proof of the following theorem gives us some intuition why, for deterministic non-cooperative systems, there are severe differences between the sequential mode and the asynchronous or the maximally parallel mode.

Theorem 2. $N_{deta}OP_1^{sequ}(ncoo, pro_{1,1}, inh_{1,1}) = NRE$.

Proof. Consider an arbitrary deterministic register machine $M = (m, B, l_0, l_h, P)$. We simulate M by a deterministic P system $\Pi = (O, \{a_1\}, l_0, R)$ where

$$
\begin{aligned}
O &= \{a_j \mid 1 \leq j \leq m\} \cup \{l, l_1, l_2 \mid l \in B\}, \\
R &= \{l \to a_j l' \mid (l : ADD(j), l') \in P\} \\
&\quad \cup \{l \to l_1|_{a_j}, a_j \to a'_j|_{l_1, \neg a'_j}, l_1 \to l_2|_{a'_j}, a'_j \to \lambda|_{l_2}, l_1 \to l'|_{\neg a'_j}, \\
&\qquad l \to l''|_{\neg a_j} \mid (l : SUB(j), l', l'') \in P\}.
\end{aligned}
$$

We claim that Π is deterministic and non-cooperative, and it accepts the same set as M. □

As can be seen in the construction of the deterministic P system in the proof above, the rule $a_j \to a'_j|_{l_1, \neg a'_j}$ used in the sequential mode can be applied exactly once, priming exactly one symbol a_j to be deleted afterwards. Intuitively, in the asynchronous or the maximally parallel mode, it is impossible to choose only one symbol out of an unbounded number of copies to be deleted. The bounding operation defined above will allow us to put this intuition into a formal proof.

3.2 Asynchronous and Maximally Parallel Systems

Fix an arbitrary deterministic controlled non-cooperative P system. Take k as the maximum of size of all multisets in all context conditions. Then, the bounding does not influence applicability of rules, and $b_k(u)$ is halting if and only if u is halting. We proceed by showing that bounding induces equivalence classes preserved by any computation.

Lemma 2. *Assume* $u \Rightarrow x$ *and* $v \Rightarrow y$. *Then* $b_k(u) = b_k(v)$ *implies* $b_k(x) = b_k(y)$.

Proof. Equality $b_k(u) = b_k(v)$ means that for every symbol $a \in O$, if $|u|_a \neq |v_a|$ then $|u|_a \geq k$ and $|v|_a \geq k$, and we have a few cases to be considered. If no rule is applicable to a, then the inequality of symbols a will be indistinguishable after bounding also in the next step (both with at least k copies of a). Otherwise, exactly one rule r is applicable to a (by determinism, and bounding does not affect applicability), then the difference of the multiplicities of the symbol a may only lead to differences of the multiplicities of symbols b for all $b \in rhs(r)$. However, either all copies of a are erased by the rule $a \to \lambda$ or else at least one copy of a symbol b will be generated from each copy of a by this rule alone, so $|x|_b \geq |u|_a \geq k$ and $|y|_b \geq |v|_a \geq k$, so all differences of multiplicities of an object b in u and v will be indistinguishable after bounding in this case, too. □

Corollary 1. *If $b_k(u) = b_k(v)$, then u is accepted if and only if v is accepted.*

Proof. Let w be the fixed part of the initial configuration. Then we consider computations from uw and from vw. Clearly, $b_k(uw) = b_k(vw)$. Equality of boundings is preserved by one computation step, and hence, by any number of computation steps.

Assume the contrary of the claim: one of the computations halts after s steps, while the other one does not, i.e., let $uw \Rightarrow^s u'$ and $vw \Rightarrow^s v'$. By the previous paragraph, $b_k(u') = b_k(v')$. Since bounding does not affect applicability of rules, either both u' and v' are halting, or none of them. The contradiction proves the claim. □

We should like to notice that the arguments in the proofs of Lemma 2 and Corollary 1 are given for the maximal parallel mode; following the observation stated at the end of Remark 4, these two results can also be argued for the asynchronous mode.

Theorem 3. *For deterministic P systems working in the asynchronous or in the maximally parallel mode, we have the following characterization:*

$$\begin{aligned} NFIN \cup coNFIN &= N_{deta}OP_1^{asyn}(ncoo, pro_{1,*}, inh_{1,*}) \\ &= N_{deta}OP_1^{maxpar}(ncoo, pro_{1,*}) \\ &= N_{deta}OP_1^{maxpar}(ncoo, inh_{1,*}) \\ &= N_{deta}OP_1^{asyn}\left(ncoo, (pro_{*,*}, inh_{*,*})_*, pri\right) \\ &= N_{deta}OP_1^{maxpar}\left(ncoo, (pro_{*,*}, inh_{*,*})_*, pri\right). \end{aligned}$$

Proof. Each equivalence class induced by bounding is completely accepted or completely rejected. If no infinite equivalence class is accepted, then the accepted set is finite (containing numbers not exceeding $(k-1) \cdot |O|$). If at least one infinite equivalence class is accepted, then the rejected set is finite (containing numbers not exceeding $(k-1) \cdot |O|$). This proves the "at most $NFIN \cup coNFIN$" part.

In Examples 1 and 2 we have already shown that

$$N_{deta}OP_1^{\alpha}(ncoo, pro_{1,*}, inh_{1,*}) \supseteq FIN \cup coNFIN$$

for $\alpha \in \{asyn, maxpar\}$ as well as

$$N_{deta}OP_1^{maxpar}(ncoo, \gamma_{1,*}) \supseteq FIN \cup coNFIN$$

for $\gamma \in \{pro, inh\}$. This observation concludes the proof. □

There are several questions remaining open: First of all, we do not know whether inhibitors in the rules are sufficient to yield $FIN \cup coNFIN$ with the asynchronous mode, too. Moreover, it would be interesting to see if the parameter K used in the proof of the preceding theorem induces an infinite hierarchy on the families $N_{deta}OP_1^{\alpha}(ncoo, \gamma_{1,K})$, $\alpha \in \{asyn, maxpar\}$, $\gamma \in \{pro, inh\}$.

4 Conclusion

We have shown that, like in case of catalytic P systems, for non-cooperative P systems with promoters and/or inhibitors (with or without priorities), determinism is a criterion drawing a borderline between universality and decidability. In fact, for non-cooperative P systems working in the maximally parallel or the asynchronous mode, we have computational completeness in the unrestricted case, and only all finite number sets and their complements in the deterministic case.

Acknowledgements. The first author gratefully acknowledges the project RetroNet by the Lombardy Region of Italy under the ASTIL Program (regional decree 6119, 20100618).

References

1. A. Alhazov, D. Sburlan: Ultimately Confluent Rewriting Systems. Parallel Multiset-Rewriting with Permitting or Forbidding Contexts. *Membrane Computing, 5th International Workshop* WMC 2004, Milano, Revised Selected and Invited Papers (G. Mauri et al., eds.), LNCS 3365, Springer, 2005, 178–189.
2. R. Freund, L. Kari, M. Oswald, P. Sosík: Computationally Universal P Systems without Priorities: Two Catalysts are Sufficient, *Theoretical Computer Science* **330**, 2, 2005, 251–266.
3. R. Freund, M. Kogler, M. Oswald, A General Framework for Regulated Rewriting Based on the Applicability of Rules. In: J. Kelemen, A. Kelemenová, *Computation, Cooperation, and Life*, Springer, LNCS 6610, 2011, 35–53.
4. O.H. Ibarra, H.-C. Yen: Deterministic Catalytic Systems are Not Universal, *Theoretical Computer Science* **363**, 2006, 149–161.
5. M.L. Minsky: *Finite and Infinite Machines*, Prentice Hall, Englewood Cliffs, New Jersey, 1967.
6. Gh. Păun: *Membrane Computing. An Introduction*, Springer, 2002.
7. Gh. Păun, G. Rozenberg, A. Salomaa: *The Oxford Handbook of Membrane Computing*, Oxford University Press, 2010.
8. G. Rozenberg, A. Salomaa: *Handbook of Formal Languages*, 3 vol., Springer, 1997.
9. P systems webpage. `http://ppage.psystems.eu`